DATA SCIENTIST PROJECT

DEVELOPING A CREDIT RISK PREDICTION MODEL FOR IDX PARTNERS

HTTPS://GITHUB.COM/DENILSONJOSHUA

CONTENT

PROBLEM STATEMENT

In this project, I will analyze the ID/X Partners default Risk machine learning to predict loan repayment using historical data. It's a standard supervised classification task with binary labels: 0 for on-time repayment and 1 for repayment difficulties.

DEFINE TARGET VALUE

```
data.loan_status.value_counts(normalize=True)*100
```

```
Current                                                48.087757
Fully Paid                                             39.619332
Charged Off                                             9.109236
Late (31-120 days)                                      1.479782
In Grace Period                                         0.674695
Does not meet the credit policy. Status:Fully Paid      0.426349
Late (16-30 days)                                       0.261214
Default                                                 0.178432
Does not meet the credit policy. Status:Charged Off     0.163205
Name: loan_status, dtype: float64
```

from `loan_status` feature can be categorized into 2 types, namely:

bad debt :

- Charged off
- Default
- Late (31-120 days)
- Does not meet the credit policy.

good debt:

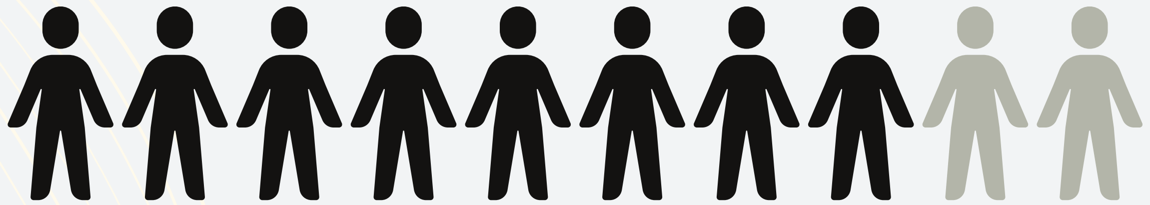
- In Grace Period
- Fully Paid
- Late (16-30 days)
- Current
- Does not meet the credit policy.

```
data['bad_flag'].value_counts(normalize=True)*100

0    89.069346
1    10.930654
Name: bad_flag, dtype: float64
```

STATISTICS

The number of individuals marked as bad loans is far less than good loans. This causes this problem to become an imbalanced dataset problem.

89%



Data Preprocessing

```
data['emp_length'].unique()
```

```
array(['10+ years', '< 1 year', '1 year', '3 years', '8 years', '9 years',
       '4 years', '5 years', '6 years', '2 years', '7 years', nan],
      dtype=object)
```

```
data['emp_length_int'] = data['emp_length'].str.replace('\+ years', '')
data['emp_length_int'] = data['emp_length_int'].str.replace('< 1 year', str(0))
data['emp_length_int'] = data['emp_length_int'].str.replace(' years', '')
data['emp_length_int'] = data['emp_length_int'].str.replace(' year', '')
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_1528\1742887160.py:1: FutureWarning:
a future version.
  data['emp_length_int'] = data['emp_length'].str.replace('\+ years', '')
```

```
data['emp_length_int'] = data['emp_length_int'].astype(float)
```

Merubah format value dari feature emp_length dan term

```
data['term'].unique()
```

```
array([' 36 months', ' 60 months'], dtype=object)
```

```
data['term_int'] = data['term'].str.replace(' months', '')
data['term_int'] = data['term_int'].astype(float)
```

```
data.drop('term', axis=1, inplace=True)
```

```
data['earliest_cr_line'].head(3)
```

```
0    Jan-85
1    Apr-99
2    Nov-01
Name: earliest_cr_line, dtype: object
```

```
data['earliest_cr_line_date'] = pd.to_datetime(data['earliest_cr_line'], format='%b-%y')
data['earliest_cr_line_date'].head(3)
```

```
0   1985-01-01
1   1999-04-01
2   2001-11-01
Name: earliest_cr_line_date, dtype: datetime64[ns]
```

```
data['mths_since_earliest_cr_line'] = round(pd.to_numeric((pd.to_datetime('2017-12-01') - data['earliest_cr_line_date']) / np.timedelta64(1, 'M')))
data['mths_since_earliest_cr_line'].head(3)
```

```
0    395.0
1    224.0
2    193.0
Name: mths_since_earliest_cr_line, dtype: float64
```

Modified earliest_cr_line, issue_d, last_pymnt_d, next_pymnt_d, dan last_credit_pull_d to calculate time elapsed since that date, using a reference date of 2017-12-01 for relevance to the dataset (2007-2014).

Handling Missing Value

```
check_missing = data.isnull().sum() * 100 / data.shape[0]
check_missing[check_missing > 0].sort_values(ascending=False)
```

```
mths_since_last_record         86.566585
mths_since_last_delinq         53.690554
tot_coll_amt                   15.071469
tot_cur_bal                    15.071469
emp_length_int                  4.505399
revol_util                      0.072917
collections_12_mths_ex_med      0.031097
delinq_2yrs                     0.006219
inq_last_6mths                  0.006219
open_acc                        0.006219
pub_rec                         0.006219
total_acc                       0.006219
acc_now_delinq                  0.006219
mths_since_earliest_cr_line     0.006219
annual_inc                      0.000858
dtype: float64
```

Di sini, kolom-kolom dengan missing values di atas 75% dibuang

columns with missing values above 75% are discarded

Handling Missing Value

```
data['annual_inc'].fillna(data['annual_inc'].mean(), inplace=True)
data['mths_since_earliest_cr_line'].fillna(0, inplace=True)
data['acc_now_delinq'].fillna(0, inplace=True)
data['total_acc'].fillna(0, inplace=True)
data['pub_rec'].fillna(0, inplace=True)
data['open_acc'].fillna(0, inplace=True)
data['inq_last_6mths'].fillna(0, inplace=True)
data['delinq_2yrs'].fillna(0, inplace=True)
data['collections_12_mths_ex_med'].fillna(0, inplace=True)
data['revol_util'].fillna(0, inplace=True)
data['emp_length_int'].fillna(0, inplace=True)
data['tot_cur_bal'].fillna(0, inplace=True)
data['tot_coll_amt'].fillna(0, inplace=True)
data['mths_since_last_delinq'].fillna(-1, inplace=True)
```

for columns whose missing value is below 75%, imputation is carried out

Feature Transformation and Scaling

```
categorical_cols = [col for col in data.select_dtypes(include='object').columns.tolist()]
```

```
onehot = pd.get_dummies(data[categorical_cols], drop_first=True)
```

```
onehot.head()
```

	grade_B	grade_C	grade_D	grade_E	grade_F	grade_G	home_ownership_MORTGAGE	home_ownership_NONE	home_ownership_OTHER	home_ownership_OWN	home_ownership_RENT
0	1	0	0	0	0	0	0	0	0	0	1
1	0	1	0	0	0	0	0	0	0	0	1
2	0	1	0	0	0	0	0	0	0	0	1
3	0	1	0	0	0	0	0	0	0	0	1
4	1	0	0	0	0	0	0	0	0	0	1

All categorical columns are done One Hot Encoding.

Feature Transformation and Scaling

```python
numerical_cols = [col for col in data.columns.tolist() if col not in categorical_cols + ['bad_flag']]
```

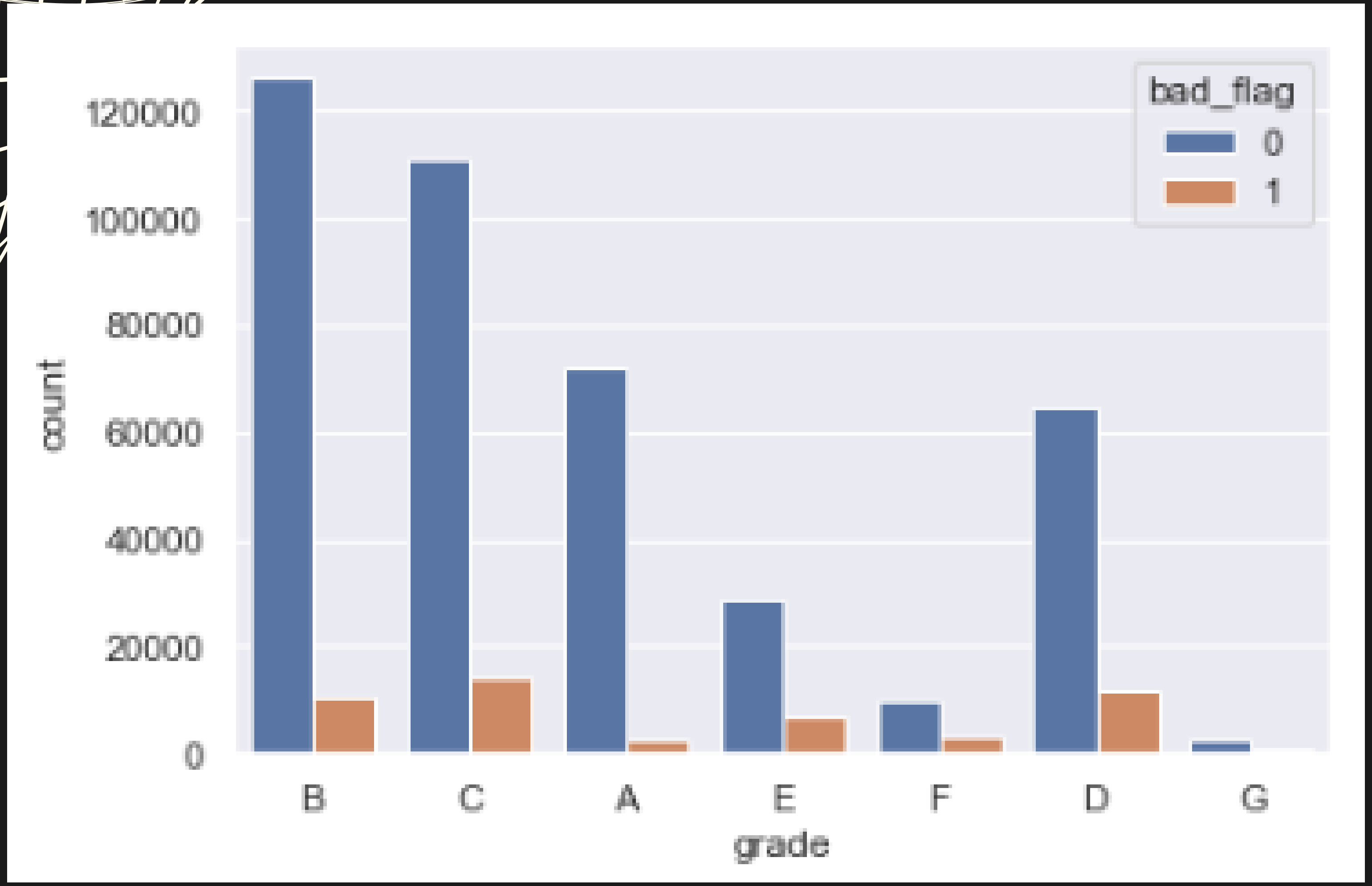
```python
from sklearn.preprocessing import StandardScaler

ss = StandardScaler()
std = pd.DataFrame(ss.fit_transform(data[numerical_cols]), columns=numerical_cols)
```

```python
std.head()
```

	loan_amnt	int_rate	annual_inc	dti	delinq_2yrs	inq_last_6mths	mths_since_last_delinq	open_acc	pub_rec	revol_bal	revol_util	total_acc	out_prncp	total_rec_late_fee	recoveries
0	-1.124392	-0.729587	-0.896551	1.328632	-0.357012	0.178920	-0.708792	-1.641166	-0.31429	-0.124888	1.159498	-1.384557	-0.693944	-0.123464	-0.154549
1	-1.426088	0.330634	-0.787387	-2.065791	-0.357012	3.843328	-0.708792	-1.641166	-0.31429	-0.703378	-1.965980	-1.815538	-0.693944	-0.123464	0.057470
2	-1.438156	0.488979	-1.110294	-1.082491	-0.357012	1.095022	-0.708792	-1.841641	-0.31429	-0.642003	1.782070	-1.298361	-0.693944	-0.123464	-0.154549
3	-0.521001	-0.077850	-0.438063	0.354248	-0.357012	0.178920	0.860811	-0.237839	-0.31429	-0.514224	-1.478018	1.028934	-0.693944	3.099264	-0.154549
4	-1.365749	-0.261438	0.122311	0.091865	-0.357012	-0.737182	0.991612	0.764538	-0.31429	0.558748	-0.094058	1.115130	-0.573268	-0.123464	-0.154549

All numerical columns are standardized using StandardScaler.

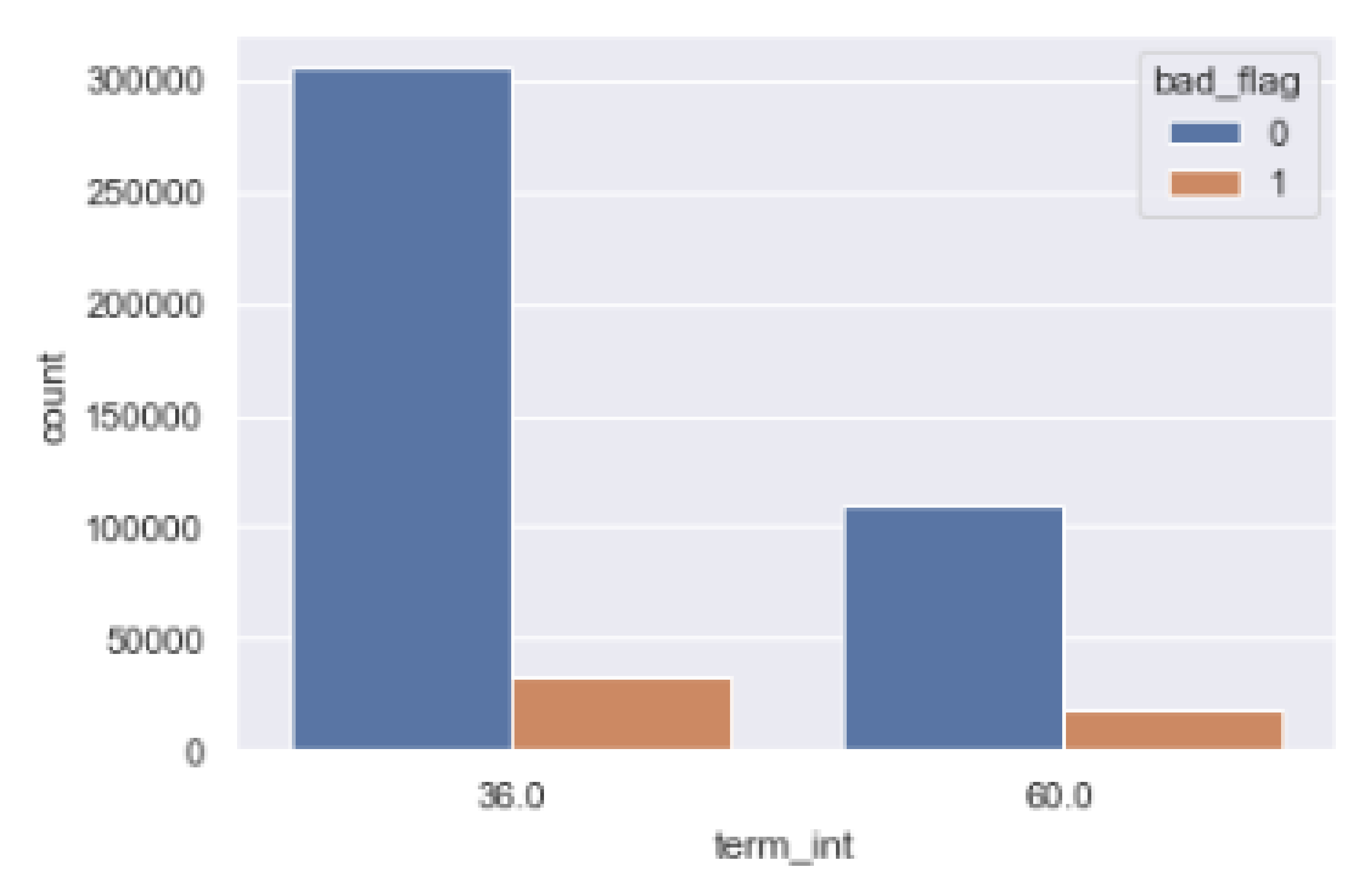
EXPLORATORY DATA ANALYSIS



Grade A, F is low risk while grade B, C, E, D has a high risk

Home Ownership with Mortgage status has a higher chance of returning, but overall the difference is not significant with OWN and RENT status

EXPLORATORY DATA ANALYSIS



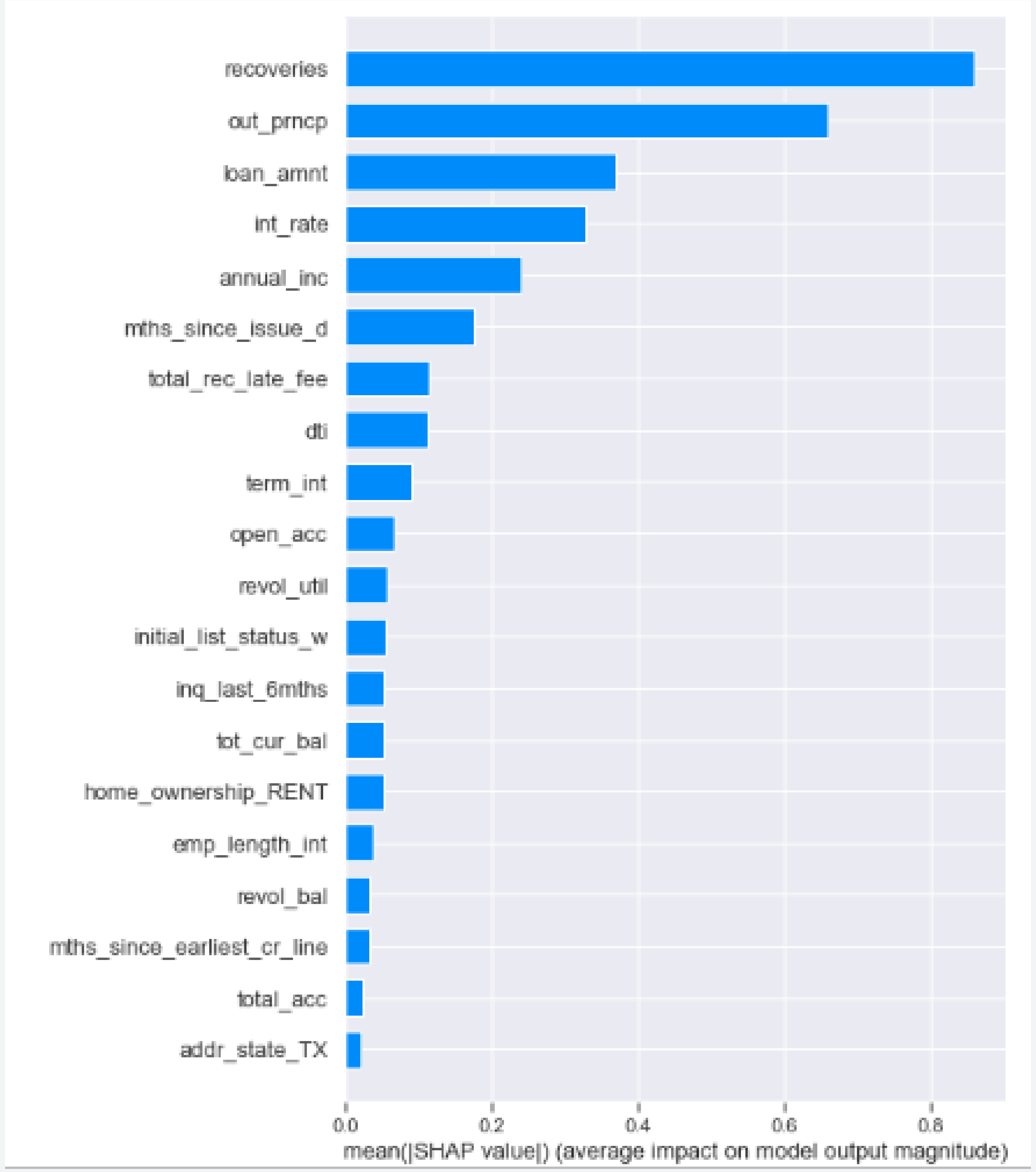
Term period 36 months has a high risk for loan money not being returned

MACHINE LEARNING IMPLEMENTATION

	Algorithm used	AUC Train	AUC Test	KS Train	KS Test
0	XgBoost	0.90	0.89	0.64	0.62
1	Random Forest	0.86	0.86	0.56	0.56
2	Decission Tree	0.85	0.85	0.56	0.56

Model yang dibangun menghasilkan performa AUC = 0.89 dan KS = 0.62. Pada dunia credit risk modeling, umumnya AUC di atas 0.7 dan KS di atas 0.3 sudah termasuk performa yang baik.

FEATURE IMPORTANCE



CONCLUSION

1. **Grade B, C, D, E Has a high risk**
2. **Term above 36 months has a high risk**
3. **Borrowers who live rented or mortage have a higher risk of default**
4. **Interest rate above 20% tends to default**

RECOMENDATION

1. When the loan is running (look last total payment amount received & outstanding principal), the company can provide the option to give financing restructuring
2. Companies should consider multiplying with interest rates belowe 20% and with a loadn period of 60 months